Trigger Discussion July 2008

Introduction

http://www-glast.slac.stanford.edu/software/core/minutes_11062007.htm

https://confluence.slac.stanford.edu/download/attachments/20011/CalDigi+variable+readout.pdf

TriggerAlg::tracker and TriRowBitsAlg

The code to compute the 3inARow condition based on digi is duplicated between these 2 pieces of code. This is even dangerous, as everything in TriggerAlg relies on the local computation which ends up in EventHeader, while it is the version in TriRowBitsAlg that actually makes it to the TDS and then to the L1 ROOT object.

What should be done :

1) reverse TriggerAlg and TriRowBitsAlg in the basicOptions.txt

2) remove the code in TriggerAlg::tracker and replace by a call to the TDS.

User defined properties should always take precedence : exemple with WindowMask

The current snippet is

```
   /// Apply window mask. Only proceed if the window was opened  /    /// or any trigger bit was set if window
open mask was not available.

    if (m_applyWindowMask){
     unsigned windowOpen=0;
     if (m_pcounter)
{ /// using ConfigSvc
/    windowOpen=trigger_bits & tcf->windowParams()->windowMask();
     }else{
    windowOpen=trigger_bits&m_mask;
     }
     if ( \!windowOpen){
    m_window_reject++;
    setFilterPassed(false);
    return sc;
     }
   }
```

m_pcounter is the pointer to the configuration service, and m_mask defaults to 0xffffffff, id est no mask applied. If I had been able to use m_mask to define the right mask to use, I could have been able to override the missing info from ConfigSvc. So, I think that a more flexible logic would be :

```
   /// Apply window mask. Only proceed if the window was opened  /    /// or any trigger bit was set if window
open mask was not available.
    if (m_applyWindowMask){
     unsigned windowOpen=0;
     if (m_pcounter&&m_mask==0xffffffff){ /// using ConfigSvc/    windowOpen=trigger_bits & tcf->windowParams()-
>windowMask();     }
}else{
windowOpen=trigger_bits&m_mask;
}
if( !windowOpen)
{
    m_window_reject++;
   setFilterPassed(false);
    *return* sc;
     }
}
```

1) ConfigSvc not available : m_mask is used
2) ConfigSvc available and mask is default pass_all : the user did not set a specific mask and the service is there, so rely on the service
3) ConfigSvc available but mask is not default : the user expresses an intent, as to what he/she wants to applied, so use his/her mask and not the one provided by ConfigSvc.